Actionable Insights in Multivariate Time-series for Urban Analytics

Anika Tabassum*, Supriya Chinthavali°, Varisara Tansakul°, B. Aditya Prakash△
*Department of Computer Science, Virginia Tech
°Oak Ridge National Laboratory
△ College of Computing, Georgia Institute of Technology
Email: anikat1@vt.edu, {chinthavalis,tansakulv}@ornl.gov, badityap@cc.gatech.edu

ABSTRACT

Multivariate time-series data are gaining popularity in various urban applications, such as emergency management, public health, etc. Segmentation algorithms mostly focus on identifying discrete events with changing phases in such data. For example, consider a power outage scenario during a hurricane. Each time-series can represent the number of power failures in a county for a time period. Segments in such time-series are found in terms of different phases, such as, when a hurricane starts, counties face severe damage, and hurricane ends. Disaster management domain experts typically want to identify the most affected counties (time-series of interests) during these phases. These can be effective for retrospective analysis and decision-making for resource allocation to those regions to lessen the damage. However, getting these actionable counties directly (either by simple visualization or looking into the segmentation algorithm) is typically hard. Hence we introduce and formalize a novel problem RaTSS (Rationalization for time-series segmentation) that aims to find such time-series (rationalizations), which are actionable for the segmentation. We also propose an algorithm Find-RaTSS to find them for any black-box segmentation. We show Find-RaTSS outperforms non-trivial baselines on generalized synthetic and real data, also provides actionable insights in multiple urban domains, especially disasters and public health.

CCS CONCEPTS

• **Information systems** → **Data mining**.

KEYWORDS

Multivariate time-series, Rationalization, Explanations, Urban analytics

ACM Reference Format:
Anika Tabassum*, Supriya Chinthavali°, Varisara Tansakul°, B. Aditya Prakash△. 2021. Actionable Insights in Multivariate Time-series for Urban Analytics. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3482410

1 INTRODUCTION

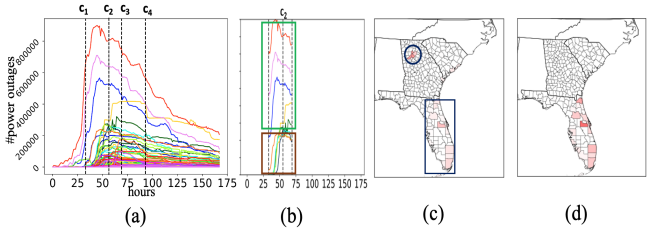


Figure 1: Disaster example (a): 2017 Hurricane Irma timeseries (representing power failures of a county during the hurricane). The first and last cut point are based on the hurricane landfall (Sept. 10) and end time (Sept. 12). Cupoints c_2 and c_3 is around the time when hurricane is changing trajectory. (b) shows a snippet and (c) a heatmap of top 10 most important counties found by our algorithm across cutpoint c_2. (d) shows a simple magnitude-based *TOIs* heatmap across c_2. Brighter colors indicate larger importance weights ((c),(d)).

Motivation: Multivariate time-series data, where each timestamp has readings from the observations of multiple entities or sensors, are prevalent in various urban scenarios, ranging from critical infrastructures, public health, and so on.

Urban domain experts such as emergency management and health care authorities segment such data for identifying meaningful events. For these events, they want to understand *time-series of interests* (*TOIs*), i.e., the signals which are more susceptible across each particular event. For example, in a hurricane disaster scenario, a set of multivariate time-series can represent power failures across a set of counties. A domain expert can identify the events which correspond to different phases of the hurricanes (e.g., based on severity of damage), by using multiple segmentation algorithms [7, 12, 21]. However, most time-series segmentation algorithms [11, 12, 21]

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482410

are usually complex, and while they give high quality segmentations, they do not give ready actionable insights to identify the *TOIs* across events.

Finding time-series of interests (*TOIs*): We next discuss how *TOIs* gives actionable insights which are useful for domain experts in context of urban analytics. For the hurricane power failure data (mentioned above) domain experts (DEs) such as emergency management authorities seek solutions to reduce power outages in different counties. If DEs get actionable insights, i.e., which time-series/counties are the most important with respect to an event, they can send personnel to fix damage and alert local authorities to reduce further loss [6]. For example, consider Fig. 1 for the 2017 Hurricane Irma (where Fig. 1(a) shows the hurricane time-series data with the segmentation, which roughly correspond to different phases). Fig. 1(b) shows the failure time-series snippet of top 10 counties (i.e. time-series) across the second cutpoint (c_2) found by our algorithm. We see some of the counties have a very high change of power failures (green box) which are obviously useful to guide resource allocation as they denote widespread problems. At the same time there are also some less obvious counties near Atlanta too (brown box in Fig 1(b)). These counties have relatively lower but a sudden change of failures compared to the other counties. Further, see the geographic heatmap plot of our *TOIs* (based on their importance weights as learnt by our algorithm) in Fig. 1(c). Clearly these counties (denoted by blue circle) are not obvious: they are located at the north-west corner, far away from where the hurricane is present (the blue rectangle, where the rest of the important *TOIs* are). However they are still important and useful for situational awareness and resource allocation. Indeed, reports [2] suggest that this is due to a separate tropical storm happening at this time. Note that figuring out these counties by just visualizing the time-series (Fig. 1(a)) across c_2 is hard (as they get buried). In fact, further if we plot a heatmap of all the counties based on their rate of change of magnitude across c_2 (Fig. 1(d)) we *cannot* recover these counties located at the middle (as their magnitudes are relatively lower).

How can we then find such *TOIs* for events? One plausible way to solve such problems might be to somehow interpret the internal change of state of the segmentation algorithm across any cut-point. For instance, one might use a recent segmentation algorithm Autoplait [21] to segment, which learns a HMM internally. Hence one way to get the *TOIs* will be to see which time-series cause the internal HMM to change across the cut-point. However it is easy to see these will only point to time-series which help explain *model behavior*, but not necessarily give actionable insights to the DE (as explained above such actionable insights are based more on the change in the time-series itself, and how similar/dissimilar these changes are to others). Additionally, DEs may end up using multiple segmentation algorithms (such as TICC [12] which uses multilayer MRF in contrast to Autoplait) for different datasets based on what constitute meaningful events for the data. Tracking the model behavior for each of these segmentation algorithms will give different *TOIs*. Further this may become too complex to the DE, who will need additional technical help to understand the technical intricacies of the segmentation algorithm to get any actionable insights.

Our Contributions: Hence, in a collaboration between computer scientists and experts at Oak Ridge National Laboratory, we aim to find a different way to get actionable insights for any segmentation algorithm. Our contributions are:

- We introduce and formalize a novel problem Rationalization for Time-series Segmentations (RaTSS) which aims to find human-friendly and actionable *TOIs* (rationalizations) for the urban experts across the associated events in terms of *constituent time-series*.
- We propose an algorithm Find-RaTSS to automatically capture the *TOIs* in a way that is flexible and works for any black-box segmentation that a DE may use.
- Finally, we evaluate performance of Find-RaTSS with baselines on both synthetic and real general and urban data. Also, we showcase how rationalizations (*TOIs*) by Find-RaTSS are actionable directive for the DEs in urban domains like disasters and disease spread.

The rest of the paper is organized as follows. We first propose and formalize our novel problem RaTSS. Next, we design an algorithm to solve RaTSS and showcase performances of our algorithm in the synthetic and real-life urban domain. Additionally, in the experiments (Sec. 4) we show how our *TOIs* can help a variety of DEs, including public health for understanding impact of interventions in the COVID-19 pandemic. We then explore the related literature on the closest works in urban analytics and time-series mining and finally conclude with the avenues for future work. Additional experiments are in the appendix [4].

2 PROBLEM FORMULATION

Notations. Suppose, we have a multivariate time-series data matrix of m sequences $\mathbf{X} = \{x_1, x_2, ...x_m\}$, where each $x_u = x_u(t_1), ...x_u(t_t)$ has t observations. We are given a set of cutpoints $C = \{c_1, c_2, \ldots, c_k\}$, each $1 \leq c_j \leq t$. For the rest of the paper we use the term rationalizations and *TOIs* as interchangeable. An overview of all the notations are in Table 1.

The main challenges for formulating such rationalization problem across a segmentation are: **P1.** We require a general framework which works for any $\mathbf{X}$ and any segmentation C given by a black-box segmentation algorithm B. For generalization, we do not know anything about B. **P2.** We need to associate constituent time-series across each cutpoint to actionable directives. Hence we propose an intermediary weighting scheme to measure importance of each time-series which can map them towards actionable directives.

Problem 2.1 (**Informal RaTSS.**). ***Given*** *a multivariate time-series* $\boldsymbol{X}$ *and a segmentation* C *for* $\boldsymbol{X}$. ***Find*** *the rationalization weight vector of size* $m \times 1$, $\boldsymbol{r}_j$ *across each cutpoint* c_j *in* C, *where each value* r_j^u *in* $\boldsymbol{r}_j$ *is a scalar and represents the importance weight for time-series* x_u *across* c_j.

What is r_j^u? One possible approach to measure importance is to assign a numerical weight r_j^u to the time-series x_u in terms of its change across a cutpoint c_j. However, we do not know anything about the segmentation algorithm B (**P1**), besides the way we should measure change (e.g., using time-series features) may be different for each cut-point. Hence feature-engineering to work for all B correctly is not possible.

Hence we present another idea: since C is selected by B among all possible segmentations, we may assume C by B is the best in some

Table 1: Notations.

Symbol	Description
$\mathbf{X} \in \mathbb{R}^{m \times T}$	Data matrix consisting of m time-series each having t timestamps
C	Segmentation with a set of cutpoints for $\mathbf{X}$
B	Any segmentation algorithm which outputs a set of cutpoints on $\mathbf{X}$.
$\mathbf{r}_j$	$m \times 1$ rationalization weight vector
$G(\mathcal{S}, \mathcal{E})$	A segment graph of $\mathcal{S}$ nodes (s_{ij}) and $\mathcal{E}$ edges (e_{ijk}) connected by weight $\mathbf{w}_{ijk}$
s_{ij}	A node (segment) of G consisting of the sub-sequence of m time-series within $i - j$ timestamps
e_{ijk}	An edge of G representing edge weight of the connected nodes s_{ij} & s_{jk}
$\mathbf{w}_{ijk}$	A $m \times 1$ weight vector for e_{ijk}, each cell is weight of x_i in e_{ijk}
K	Total number of paths in G
$F(s_{ij})$	A function to return a $m \times f$ matrix for each time-series in s_{ij} having f features
P_B	The path in G which maps to C
P_{rest}	All possible paths in G except the path π_B
$\boldsymbol{\pi}_B$, $\boldsymbol{\pi}_{\text{rest}}$	Cost of path P_B, P_{rest}
$\boldsymbol{\alpha}$	Global latent weight vector
λ_1, λ_2	Scalar hyper-parameters used for rationalization Eq. 2

sense (as after all the algorithm B output C as the segmentation, choosing it over *all the other possible segmentations*). It is natural to assume that the rationalization weight vectors $\mathbf{r}_j$ should be set in a way that explains *why* C becomes better compared to all other possible segmentations. Hence our main idea is to consider C as the best compared to other possible segmentations, and choose $\mathbf{r}_j$s to make it so.

Segment Graph. We want to efficiently represent all possible segmentations (as they will be exponential in number to the length of the sequence). Recently [7] proposed a 'segment graph' data structure G to represent $\mathbf{X}$ and its segmentations efficiently. Hence we leverage this data structure and convert our rationalization problem in terms of a graph optimization over G. The segment graph $G(\mathcal{S}, \mathcal{E})$ is a *Directed Acyclic Weighted Graph* (DAWG) consisting of a set of nodes $\mathcal{S}$ and set $\mathcal{E}$ of edges. Next, we define its components.

*Definition 2.1 (**Node set of** G).* The node set $\mathcal{S}$ consists of all possible segments in X. That is, node s_{ij} in node set $\mathcal{S}$ of G consists of the subsequence of $\mathbf{X}$ of consecutive timestamps i to j. Additionally $\mathcal{S}$ consists of s_o and t_a, two dummy nodes to represent the start and end of the time-series.

*Definition 2.2 (**Edge set of** G).* An edge e_{ijk} in edge set $\mathcal{E}$ connects two adjacent segments s_{ij} and s_{jk}, where $i < j < k$ and $1 \leq i < T-1$, $1 < k \leq T$. Each e_{ijk} is mapped to a possible cutpoint c_j a timestamp t_j in $\mathbf{X}$.

The edge weight vector $\mathbf{w}_{ijk}$ for the corresponding e_{ijk} represents the 'change' of every x_u between the adjacent segments of any cutpoint c_j. Note that all segmentations of $\mathbf{X}$ (including the segmentation C) naturally get mapped to paths in G. We further define K (the total number of segmentations) as the total number of possible paths from s_o to t_a paths in G.

*Definition 2.3 (**Segmentation** C **in** G).* The path P_B in G is the path from s_o to t_a s.t. each edge $e_{ijk} \in P_B$ is mapped to the corresponding $c_j \in C$.

*Definition 2.4 (**Other possible segmentations**).* P_{rest} is the set of all other possible paths in G from s_o to t_a other than P_B. Every path $p_v \in P_{\text{rest}}$ represents a possible segmentation of X.

Thus, in terms of G our idea can be stated as that P_B should be the best path from s_o to t_a in G (compared to all paths in P_{rest}). Further, $\mathbf{r}_j$ should be set in a way that helps make P_B the best path. Clearly the weight vectors $\mathbf{w}_{ijk}$ (which represent how different adjoining segments are across each edge) also should play a role in quantifying the quality of each path.

Hence, we need a 'quality' metric *Quality* to compare the paths which should depend on the rationalization weights, the paths and the weights over the edges in the paths. Then our required condition can be stated as: $Quality(P_B, \mathbf{w}_{ijk} \text{ for each } e_{ijk} \in P_B, \mathbf{r}) > Quality(p_v, \mathbf{w}_{ijk} \text{ for each } e_{ijk} \in p_v, \mathbf{r}), \forall p_v \in P_{\text{rest}}$. It is not clear how $\mathbf{r}$ which has been defined only over P_B can affect the quality of other paths. One way to handle this issue is to create 'rationalizations' over all edges - which will lead to severe over-parameterization as the number of edges in G is $O(T^3)$. Hence, we propose to instead have a *global latent* weight vector $\boldsymbol{\alpha} \in \mathbb{R}^{m \times 1}$ whose magnitude captures the global latent importance of each time-series, and then use $\boldsymbol{\alpha}$ over the edges in P_B to get $\mathbf{r}_j$ for each cutpoint $c_j \in C$. Our condition can be re-written as follows:

$$Quality(P_B, \alpha) > Quality(p_v, \alpha), \forall p_v \in P_{\text{rest}} \tag{1}$$

with $\mathbf{r}_j = \text{someFunctionOf}(\alpha, \mathbf{w}_{ijk})$.

We now informally state our problem in terms of a graph-based framework.

Problem 2.2 (**Informal Graph based RaTSS.**). ***Given, X**, C, and a segment graph G on **X**, so that its path P_B corresponds to C. **Find** $\boldsymbol{\alpha}$ to satisfy Eq. 1 and generate the rationalization weight vector $\mathbf{r}_j$ across each edge e_{ijk} in P_B.*

Formalizing Problem 2.2. To formalize this problem, several questions arise: **Q1.** *What is Quality?* Due to **P1**, we cannot consider any prior knowledge about the segmentation algorithm while designing the quality metric. A possible approach is to assume *Quality* as *cost* of the path. Hence, we simply consider the *weighted length* of the path as its cost. Note that, the length of the path will intuitively measure how different its adjacent segments are. Intuitively, segmentation algorithms try to find cutpoints so that each segment corresponds to a distinct phase in some sense, and hence adjacent segments are expected to be very 'different' from one another. In other words, in our framework, we want to learn $\boldsymbol{\alpha}$ which makes P_B the *longest* weighted path in G from s_o to t_a.

*Definition 2.5 (**Length of a path**).* The length $\boldsymbol{\pi}_v$ of a path p_v is an $m \times 1$ vector, each component represents the sum of all the edge weights $\mathbf{w}_{ijk}$ in p_v for time-series x_u. Hence, $\boldsymbol{\pi}_v = \sum_{e_{ijk} \in p_v} \mathbf{w}_{ijk}$.

Hence we can now write the *Quality* function for any path p_v (i.e. the cost of p_v) as follows:

$$Quality(p_v, \boldsymbol{\alpha}) = \sum_{e_{ijk} \in p_v} \boldsymbol{\alpha}^T \mathbf{w}_{ijk} = \sum_{e_{ijk} \in p_v} \sum_{u=1}^{m} \boldsymbol{\alpha}_u \mathbf{w}^u_{ijk}$$

Q2. *How to generate $\mathbf{r}_j$ for each c_j using $\boldsymbol{\alpha}$?* As we discussed before (in Eq. 1), $\mathbf{r}_j$ = someFunctionOf(α,$\mathbf{w}_{ijk}$). $\mathbf{w}_{ijk}$ represents the change across an edge, while $\boldsymbol{\alpha}$ gives us the global latent weight for each time-series, Additionally, recall that $\mathbf{r}_j$ is intuitively the importance of each time-series over every cut-point $c_j \in C$. Hence a simple way to get $\mathbf{r}_j$ is $\mathbf{r}_j = |\alpha \odot \mathbf{w}_{ijk}|$, where $\odot$ is the standard element-wise vector dot product (and the modulus is because $\boldsymbol{\alpha}$ can be negative). **Q3.** *How to set $\boldsymbol{w}_{ijk}$?* Note that we still haven't precisely defined $\mathbf{w}_{ijk}$. As discussed before $\mathbf{w}_{ijk}$ just should reflect some change in each time-series across the edge. Keeping in mind **P2**, we just represent each node s_{ij} as a $m \times f$ feature matrix where $F(s_{ij}) = [\mathbf{f}^1_{ij}, ...\mathbf{f}^m_{ij}]$, each $\mathbf{f}^u$ is a $f \times 1$ feature vector of time-series x_u in segment s_{ij}. We can set $\mathbf{w}_{ijk} = ||F(s_{ij}) - F(s_{jk})||_{1,2}$.

This is not feature-engineering. We choose only basic statistical features $\mathbf{f}$ (mean, variance, minimum and maximum in this paper), as we only need to capture basic changes across the segments - not how the segmentation algorithm B regards as the change (a feature ablation test in the appendix [4] shows all these features are necessary and useful for any segmentation). This is different than directly choosing features to set $\mathbf{r}_j$, as now the rationalization weights will need to best explain *why* the cut-point is present, not *how* the time-series change.

Putting everything together. We next formalize our task as an optimization problem. Note that Eq. 1 will result in one inequality for each path in G, which are exponential in number (to T). Hence directly using Eq. 1 is infeasible. Instead, we tackle a simplified version, by just adding all the inequalities to get a consolidated objective. Let $\boldsymbol{\pi}_B$ be the length of path P_B. We also define $\boldsymbol{\pi}_{\text{rest}}$ as follows.

*Definition 2.6 (**Total length of** P_{rest}).* $\boldsymbol{\pi}_{\text{rest}}$ is an $m \times 1$ vector, each component represents the length of all other paths in P_{rest} other than P_B over each time-series. Hence, $\boldsymbol{\pi}_{\text{rest}} = \sum_{p_v \in P_{\text{rest}}} \boldsymbol{\pi}_v = \sum_{p_v \in P_{\text{rest}}} \sum_{\substack{e_{ijk} \in p_v \\ e_{ijk} \notin P_B}} \mathbf{w}_{ijk}$.

We can also rewrite $\boldsymbol{\pi}_{\text{rest}}$ as $\sum_{e_{ijk} \in \mathcal{E} - P_B} p_{ijk} \mathbf{w}_{ijk}$ where p_{ijk} is the number of paths in G passing through e_{ijk}. Suppose, $\boldsymbol{\Delta\pi} = \sum_{p_v \in P_{\text{rest}}} \boldsymbol{\pi}_B - \boldsymbol{\pi}_v = (K-1)\boldsymbol{\pi}_B - \boldsymbol{\pi}_{\text{rest}}$, where K is the total number of paths in G. Hence the set of inequalities in Eq. 1 will imply maximizing $\boldsymbol{\alpha}^T \boldsymbol{\Delta\pi}$. Therefore, we formalize RaTSS as follows.

Problem 2.3 (**Formal Graph based RaTSS.**). ***Given, $\boldsymbol{X}$, C,*** *a segment graph G on $\boldsymbol{X}$, so that its path P_B corresponds to C, and a Function $F(.)$ to represent a node s_{ij} on G in $m \times f$ feature matrix.* ***Find $\boldsymbol{\alpha}$*** *that satisfy* Quality$(P_B, \alpha) >$ Quality$(p_v, \alpha), \forall p_v \in P_{\text{rest}}$ *and generate rationalization $\mathbf{r}_j$, for each edge e_{ijk} in P_B such that*

$$\begin{aligned} &\arg\max_{\boldsymbol{\alpha}} \boldsymbol{\alpha}^T(\boldsymbol{\Delta\pi}) - \lambda_1 ||\boldsymbol{\alpha}||_1 \\ &\text{subject to } \boldsymbol{\alpha} \neq 0, ||\boldsymbol{\alpha}||_2^2 = 1 \end{aligned} \tag{2}$$

$$\mathbf{r}_j = |\alpha \odot \mathbf{w}_{ijk}| \tag{3}$$

Details: We want $\boldsymbol{\alpha}$ to assign a higher weight for the time-series with high value in $\boldsymbol{\Delta\pi}$ (each $\boldsymbol{\alpha}_u$ represents a weight for time-series x_u). The term $\lambda_1 ||\boldsymbol{\alpha}||_1$ is to encourage sparsity for simple explanations. The constraint $||\boldsymbol{\alpha}||_2^2 = 1$ is to ensure that $\boldsymbol{\alpha}_u$s are bounded and comparable. An overview of RaTSS is shown in Fig. 2

Remark 1: A silent assumption in RaTSS is that the segmentation (events) is meaningful to the DE. For example, when the time-series is constant, then rationalizations (*TOIs*) found by RaTSS may not be meaningful. Indeed, arguably, it is not clear in this case if there are any meaningful *TOIs* in the first place (intuitively, for homogeneous data, cut points are not useful).

Remark 2: We define our rationalizations in terms of constituent time-series only. As we discussed in Sec. 1 and show in our experiments later, such a definition is already meaningful in an urban analytics context (e.g., actionable counties in the hurricane example). However, there can be situations where a *group* of time-series can also be actionable. We plan to consider such extensions in future work.

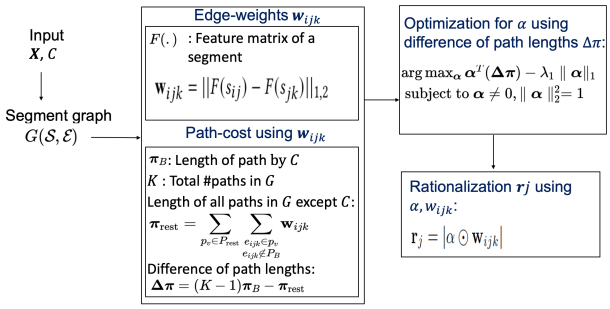


Figure 2: Overview of RaTSS

3 OUR ALGORITHM

We aim to design an efficient approach for Problem 2.3 to identify the *TOIs* for RaTSS. Our main steps are- (A) Computing $\boldsymbol{\Delta\pi}$ and (B) Optimizing $\mathbf{r}_j$. Next, we discuss in detail about these steps, present an overall algorithm Find-RaTSS and its complexities and implementation.

(A) Computing $\boldsymbol{\Delta\pi}$. We construct the segment graph $G(\mathcal{S}, \mathcal{E})$ to compute $\boldsymbol{\Delta\pi}$ from G. The computation is of three steps: (i) Computing the length of P_B, i.e. $\boldsymbol{\pi}_B$, which is trivial, (ii) Calculating the total length of all the possible paths, except $\boldsymbol{\pi}_B$ in G, i.e., $\boldsymbol{\pi}_{\text{rest}}$, and (iii) Calculating the total number of all possible paths in G, i.e., K. Our main challenge is to efficiently compute (ii), and (iii) since the number of paths in G is exponential. Next, we discuss these steps (ii) and (iii).

For computing $\boldsymbol{\pi}_{\text{rest}}$, we design an efficient technique to calculate the number of paths p_{ijk} passing through an edge e_{ijk} in constant

time, by exploiting the properties of G. Next, we show how we find p_{ijk} with the help of Lemma 3.1.

Lemma 3.1 (Number of paths through an edge e_{ijk} is p_{ijk}). *Given, i and k be the begin and end timestamp of the adjacent segments s_{ij} and s_{jk} of e_{ijk}. For total timestamp T, the number of paths through e_{ijk} is,*

$$p_{ijk} = \begin{cases} 1 & i = 1, k = T \\ 2^{(i-2)} & k = T \\ 2^{(T-k-1)} & i = 1 \\ 2^{(i-2)+(T-k-1)} & otherwise \end{cases} \quad (4)$$

Proof. Following is the proof for different i, j, and k.

- **Case** $i = 1, k = T$: Trivial. Since, there is only on segment from 1 to T and only one path.
- **Case** $i = 1, k < T$: (By induction). Suppose, $T = k + 1$, then the number of paths is $2^{k+1-k-1} = 2^0 = 1$. This is trivial, because from s_{jk} there is only one possible edge $s_{jk} - s_{k(k+1)}$ and thus there is only one possible path.
  Suppose, with $T = k + q$ where $q > k$, the number of paths is 2^{q-1}. We need to show, with $T = k + q + 1$, the number of paths become 2^q. Since $q + 1 > q$, all the paths that can be possible to reach from s_{jk} to $s_{j(k+q)}$ can also be possible to reach from s_{jk} to $s_{j(k+q+1)}$ using the edge $s_{j(k+q)} - s_{j(k+q+1)}$. That is, 2^{q-1} paths possible from s_{jk} to $s_{j(k+q+1)}$ using the edge $s_{j(k+q)} - s_{j(k+q+1)}$.
  Again, by the edge construction property of G, a path ends when end timestamp of a node is $T = k + q + 1$. If we remove the timestamp $k + q$ and consider only $k, k + 1, \ldots, k + q - 1, k + q + 1$, then the number of possible paths become 2^{q-1} to reach $k + q + 1$ (by induction). Because, all edges that used nodes ended with $k + q$ now can use nodes ended with $k + q + 1$.
  Thus, the total number of possible paths with and without using the edge $s_{j(k+q)} - s_{j(k+q+1)}$ becomes $2^{q-1} + 2^{q-1} = 2^{q-1+1} = 2^q$ (proved).
- **Case** $i > 1, k = T$**:** Following the above if we consider $T = i$ and the $k = 1$. The number of paths is $2^{T-k-1} = 2^{i-1-1} = 2^{i-2}$.
- **Case** *otherwise***:** Hence, if there are 2^{i-2} paths possible to reach a node s_{ij} and 2^{T-k-1} possible paths to reach timestamp T from a node s_{jk}. The total number of paths possible through the edge $s_{ij} - s_{jk}$ or e_{ijk} is $p_{ijk} = 2^{(i-2)+(T-k-1)}$.

□

Next, using Lemma 3.1, we can further compute the following corollary on the total number of paths K, intuitively, by setting $i = 1$, and $k = 2 \ldots T - 1$.

Corollary 1 (**Total number of possible paths in segment graph** G **is** K). *Given total timestamp is T, and number of paths through an edge e_{ijk} is p_{ijk}. If we consider all the edges in edge-set G as $\mathcal{E}$, the total number of possible paths in G is- $K = 1 + \sum_{i=1}^{T-1}\sum_{k=2}^{T-1} p_{ijk} = 2^{T-2}$*

Proof. According to the path property of G, every path starts from $i = 1$ and ends at timestamp T. For $i = 1, k = T$ there is only one path possible (the whole segmentation). For $1 < j < T - 1$, the first node of all other paths in G has to start from some s_{1j}. Consider $j = k$, using Lemma 3.1 we find p_{1kk}. Thus, $K = 1 + \sum_{k=2}^{T-1} p_{1kk} = 1 + 2^{T-2} - 1 = 2^{T-2}$. □

(B) Optimizing $\mathbf{r}_j$. To add the constraint to the objective of Eq. 2, we use Lagrange multiplier λ_2. We solve Eq. 2 using gradient descent learning, considering the gradient $-\Delta\boldsymbol{\pi} + \lambda_1 sign(\boldsymbol{\alpha}) + \lambda_2\boldsymbol{\alpha}$. To find hyper-parameters λ_1, λ_2 we use gridsearch technique and principally choose $\boldsymbol{\alpha}$ for which L2 norm (second term in Eq. 2) is closer to 1. We calculate the importance weight $\mathbf{r}_j$ for an edge e_{ijk} in $\boldsymbol{\pi}_B$ using Eq. 3. To select the most important time-series at every e_{ijk}, we sort $\mathbf{r}_j$ in descending order and principally select the top k time-series, when cumulative fraction of their total rationalization weight >= 0.95.

Algorithm Find-RaTSS. We next give the pseudo-code for our Algorithm 1.

Input: X: a set of time-series, C: a segmentation
Result: $\mathbf{r}_j = \{r^1, ..., r^m\}$, rationalization weight of **X** for every c_j in C
Consider F: a function for characterizing time-series. Given a segment, it returns a feature matrix $m \times f$
1. Construct nodes in segment Graph G
2. Construct edges in G
3. **foreach** $e_{ijk} \in \mathcal{E}$ **do**
   Calculate total number of paths p_{ijk} in e_{ijk} using Lemma 3.1
   Calculate edge weight $\mathbf{w}_{ijk} = ||F(s_{ij}) - F(s_{jk})||_{1,2}$
   Edge cost: $\boldsymbol{\pi}^{ijk}_{\text{rest}} = p_{ijk}\mathbf{w}_{ijk}$
end
4. $\boldsymbol{\pi}_{\text{rest}} = \sum_{e_{ijk}\in\mathcal{E}} \boldsymbol{\pi}^{ijk}_{\text{rest}}$
5. Compute $\boldsymbol{\pi}_B$
6. $K = 2^{T-2}$
7. Solve $\boldsymbol{\alpha}$ using Eq. 2, hyper-parameters λ_1, λ_2
8. **foreach** $c_j \in C$ **do**
   Compute $\mathbf{r}_j$ using Eq. 3
end

Algorithm 1: Algorithm Find-RaTSS.

Implementation Details. For faster computation and handling large data for Algorithm 1, we adopt several techniques, such as (i) parallelization, (ii) floating point precision, and (iii) normalization.

(i) *Parallelization:* For efficient and fast computation of $\boldsymbol{\pi}_{\text{rest}}$, we parallelize Algorithm 1. We divide the task of calculating number of paths for edges among a set of processors n. Also we use a shared memory from Python Multiprocessing library to compute $\boldsymbol{\pi}_{\text{rest}}$.

(ii) *Floating point precision:* To efficiently store large value of p_{ijk} and K (for $T > 900$), we rearrange Eq. 2 as $\frac{\Delta\boldsymbol{\pi}}{K} = \boldsymbol{\pi}_B - \frac{\boldsymbol{\pi}_{\text{rest}}}{K}$. And we ignore $\frac{\boldsymbol{\pi}_{\text{rest}}}{K}$ for a very small value.

(iii) *Normalization of* $\Delta\boldsymbol{\pi}$*:* For efficient optimization and better $\boldsymbol{\alpha}$ convergence, we normalize on $\frac{\Delta\boldsymbol{\pi}}{K}$ by their max value and rearrange as $\Delta\boldsymbol{\pi} = \frac{\Delta\boldsymbol{\pi}}{KM}$, $M = \max(\frac{\Delta\boldsymbol{\pi}}{K})$.

Lemma 3.2 (**Time and Space Complexity**). *The worst-case time complexity of our algorithm is as follows:*

- *Case serial:* $O(T^3mf) + O(I) + O(Cmf)$, *I= number of iterations in the gradient descent phase.*
- *Case parallel:* $O(\frac{T^3}{n}mf) + O(I) + O(Cmf)$, n = *number of processors.*

The space complexity of Find-RaTSS is $O(m + mn)$.

Proof. We discuss separate proof for time and space complexity.
Time complexity: (i) *Case serial:* The first term is to calculate π_{rest} for each e_{ijk}. Total e_{ijk} in G is $O(T^3)$. The second term is to solve α using Gradient Descent learning, and the third term is to solve $\mathbf{r}_j$ for every cutpoint c_j.

(ii) *Case Parallel:* Parallelization can be applied in Find-RaTSS for π_{rest} computation, then the total time complexity will be distributed among n processors and hence this is $O(\frac{T^3}{n}mf)$. However, time complexity of gradient descent and $\mathbf{r}_j$ computation is similar as in serial cases.
Space complexity: (i) Case serial: The first term $O(m)$ is to store π_{rest} and π_B. Since we need $O(m)$ π_{rest} and $O(m)$ π_B for m time-series. Hence $O(m) + O(m) = O(2m) \approx O(m)$.

(ii) Case parallel: To store final computation of π_{rest} and π_B it takes $O(m)$. Whereas the second term $O(mn)$ is to store temporary $\boldsymbol{\pi}_{\text{rest}}$ computation for each processor among n processors, $O(mn)$ is the number of possible edges in a segment graph G. Hence temporary space for n processor and final computation of $\boldsymbol{\pi}_B$ and $\boldsymbol{\pi}_{\text{rest}}$ is $O(mn) + O(m) = O(m + mn)$. □

Find-RaTSS is clearly linear in the number of time-series m. Although the serial version is also cubic in T, in practice we found that we were able to run the parallel version easily for all our datasets and it was quadratic in terms of T.

4 EXPERIMENTS

We implement RaTSS in Python and Matlab. All codes and datasets used in the paper have been released for research purposes[5]. Our experiments were conducted on a 4 Xeon E7-4850 CPU with 512 GB of 1066Mhz main memory. We collect both general data and domain specific urban data, to quantitatively and qualitatively evaluate the performance of Find-RaTSS. The detail description of the datasets with ground-truth (GT) rationalizations are discussed here. An overview of the datasets and the segmentation algorithm (B) used on each dataset are shown in Table 2. We use B which gives the most meaningful cutpoints (based on GT or historical events).

Table 2: Datasets Used.

SI #	Dataset	Time stamps	Time series	Cut points	Black box (B)	GT
1)	*Gaussian*	350	8	3	[12]	✓
2)	*Insect*	5000	4	2	[11]	✓
3)	*Chicken Dance*	322	4	7	[21]	✓
4)	*Great Barbet*	2200	2	2	[11]	✓
5)	*Sudden Cardiac*	7000	2	3	[11]	✓
6)	*Wikipedia*	803	3000	3	[12]	
7)	*Hurricane Harvey/Irma*	264/169	250/271	3/4	[23]	
8)	*COVID-19*	53	104	11	state emergency date[1]	
9)	*Diptheria/TB*	52/41	90/60	4/4	[12]	

General datasets: We use a variety of datasets (both synthetic and real) where we infer the ground-truth *TOIs* (Table 2 SI 1-6).

(1) *Gaussian*: We generate a synthetic data, where each time-series is a univariate Gaussian. We select three cutpoints and change the parameter of the Gaussian for specific $2-3$ time-series at those cutpoints. Ground-truth (GT) rationalizations are the time-series whose Gaussian parameters change at a cutpoint.
(2) *Insect* [11]: EPG recording of insect vector feeding, where each time-series represents an insect. Each cutpoint is an event when feeding state of the insect changes and GT is the insect whose feeding state changes at that event.
(3) *ChickenDance* [21]: Motion capture sequences of a set of sensors (left-right hands/legs) in a chicken dance originally collected by CMU[2]. An event occurs, when is a change of dance state, e.g., wings, tail feather, etc. GT are the set of sensors which have high change of motion while changing a dance state.
(4) *Great Barbet* [11]: Voice recordings of same species Barbet birds in MFCC format. Each recording is a mixture of two different birds with approximately half-a-minute snippet of their song. The cutpoints are set of events when snippet of one bird ends and other starts. GT are the set of birds, whose call ends or starts at an event.
(5) *Sudden Cardiac* [11]: ECG channel reading of hearts of patients. Events occur when heart state changes, e.g., normal heart to sudden heart failure or contraction. GT are the patient whose heart activity changes at an event.
(6) *Wikipedia*: Web traffic count of various Wikipedia articles collected daily from the July 2015- October 2017 [1].

Domain specific urban datasets: Next, we describe our domain specific datasets from urban analytics. Here as there is no ground truth, we discuss the qualitative performance our algorithm.

(7) *Hurricane Outage:* Oak Ridge National Laboratory (ORNL) has developed situational awareness tool *EAGLE-I* to collect power outage distribution of all the customers from utility websites every 15 minutes. We collect this power outage distribution of different counties for Hurricanes *Irma* and *Harvey* in the hurricane-affected areas used by [23]. Rationalizations on such data can help DE with retrospective analysis for emergency management planning.
(8) COVID-19 : COVID cases of different states are collected daily from Jan-May by New York Times[3]
(9) *Diptheria* and *TB*: Diptheria and Tuberculosis (TB) cases collected bi-weekly from the year 1900-2014 by Project Tycho[4]. Time-series are different cities of US representing Diptheria

[2] http://mocap.cs.cmu.edu
[3] https://github.com/nytimes/covid-19-data
[4] https://www.tycho.pitt.edu/

(TB) cases over time period (years). We run Dynammo [19] to replace the missing values of original data.

Baselines: We compare Find-RaTSS against plausible approaches. Our intuition is to find out the effect, by considering each cutpoint independent of the overall segmentation and without principally considering any optimization. Following we describe each baseline model.

(i) *Feature-based*: We calculate the change across every c_j using basic features, i.e., w_{ijk} and select x_u which have high w^u_{ijk}.
(ii) *Magnitute-based*: We calculate rate of change of time-series values across c_j for a window (5% of the timestamp). Our intuition is to pick the time-series easily figured out by visualization.
(iii) *Forecast error-based*: For every time-series x_u, we train an LSTM forecast model based on the segment before c_j and test the model for the segment after c_j. We select the time-series whose forecasting error is high. Since high forecasting error denotes high measure of unpredictability on the time-series magnitude after the cutpoint. In other words change of time-series is high across the cutpoint.

We do not compare against a baseline method of interpreting the internal change of state of segmentation algorithm (as described in Sec. 1) since our algorithm is general for a black-box segmentation, whereas each dataset is a practitioner of a different segmentation algorithm.

4.1 Quantitative evaluation

We compare Find-RaTSS with the baselines on the ground-truth data. For rationalizations we select top k as the maximum number of GT in the segmentation. Table 3 shows F1-measures of all the baselines mentioned above. From the table, we observe that Find-RaTSS consistently performs better than all the baselines (upto 41%).

Table 3: F1-scores of Find-RaTSS and baselines on the datasets with ground-truth

Dataset	Find-RaTSS	Feature	Magnitude	Forecast
Gaussian	**1.0**	0.27	0.42	0.17
Insect [11]	**0.83**	0.33	**0.83**	0.5
Chicken Dance [21]	**0.86**	0.81	0.71	0.5
Great Barbet [11]	**1.0**	**1.0**	0.5	0.5
Sudden Cardiac [11]	**1.0**	0.67	0.67	0

4.2 Case-Studies in domain-specific urban data

Next, we show *TOIs* by Find-RaTSS in hurricane power failures, COVID-19 interventions, and epidemiology. Additionally to show that our algorithm correctly captures rationalizations even for a large number of time-series (3000), we provide results on Wikipedia articles data.

Hurricane *Harvey*: We already explained how our rationalizations are actionable for cutpoint c_2 (See Fig. 1 in Sec. 1 for detail). The segmentation of Harvey is given during hurricane landfall

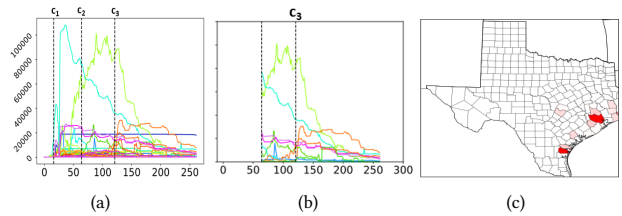
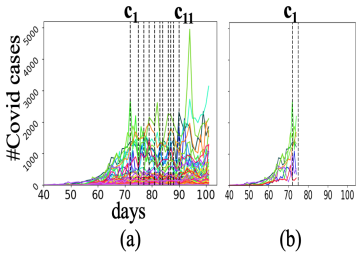


Figure 3: (a) 2017 Hurricane Harvey time-series. (b) A snippet of our rationalizations r_j for cutpoint c_3 and (c) Heatmap plot of r_j for c_3. Find-RaTSS finds non-obvious rationalizations separate from Hurricane trajectory (see detail in the text).

(around Aug. 26), change of hurricane trajectory((around Aug. 28), and end time of hurricane (around Aug. 30). Fig. 3(b) shows our rationalizations for cutpoint c_3. Note that, this is the cutpoint when hurricane is ending. Along with the decrease of the power failures of other counties, Find-RaTSS correctly highlights sudden increase of power failures of Orange, Jefferson, Hardin (Fig. 3(c) South-east corner). However, the main reason of this increase is due to rising water of Neches river due to which city lose service from major pump stations [3]. Note that, finding these non-obvious counties by visualizing the time-series (Fig. 3(a)) across the cutpoint when power failures of all other counties are decreasing is hard. These *TOIs* can help DEs prioritize resource allocation for quick recovery.

Both the hurricane power failure datasets, and the segmentation on these datsets are collected from the same source [23]. Our non-trivial rationalizations are very similar with the results provided by [23] even though the segmentation is black-box.



Rationalizations for c_1	Intervention on Jan 19
CA, OK	State emergency
MA, LA	School closure
MI	Gathering>500 ban
CT	Restaurant closed

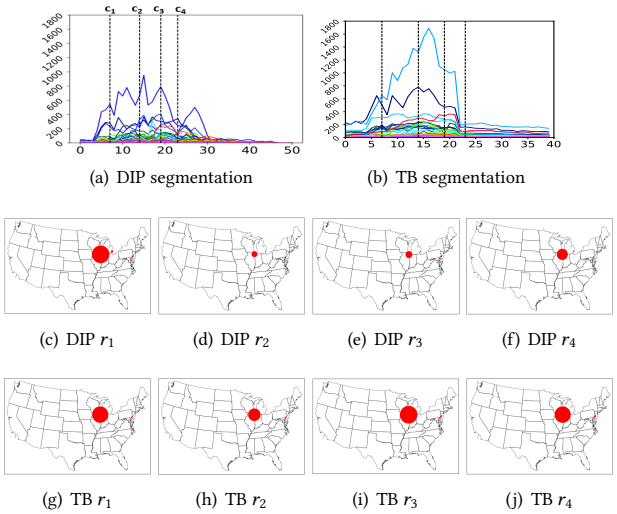
(c)

Figure 4: Covid-19 pandemic: (a) Time-series and segmentation (interventions). (b) Majority of the states for the first cutpoint c_1 inferred by Find-RaTSS had interventions in the past 2 weeks. (c) Types of interventions that happened before 2 weeks.

COVID-19 Interventions: For the current COVID-19 pandemic, our goal is to extract which states had interventions (like school closures, etc.) using Find-RaTSS and the disease trajectories. We collect (https://covidvis.github.io/) daily COVID incidences from Jan-early May and consider cutpoints as the state emergencies after 2 weeks (mean incubation period of COVID is 2-14 days). For c_1 (Fig. 4(b)), Find-RaTSS infers all the states which had some intervention around 2 weeks back (Jan 19). Fig. 4(c) shows an example of different interventions happened for the rationalizations across c_1 before 2 weeks. Overall, across all cutpoints, Find-RaTSS infers

87% states with interventions >= 2 weeks. For most rationalized states (60%), the interventions happened exactly 2 weeks back. This is very useful as in real-time it is very hard to have a complete knowledge of interventions, e.g., indeed, there is no centralized database of such acts. Hence epidemiologists need to use indirect methods (knowing which interventions are in place is crucial for modeling the disease spread). Find-RaTSS can potentially give such a method to the epidemiologists to direct attention to such states.

Diptheria (DIP): For Diphtheria and TB segmentation, we consider the time period from 1900 to 1950. This segmentation includes some historical significant events, i.e., campaign (c_1), outbreak (c_3), and invention of vaccination (c_4)[5].



(a) DIP segmentation (b) TB segmentation

(c) DIP r_1 (d) DIP r_2 (e) DIP r_3 (f) DIP r_4

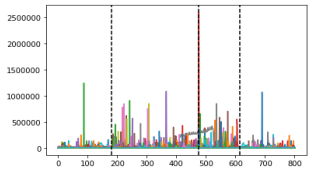
(g) TB r_1 (h) TB r_2 (i) TB r_3 (j) TB r_4

Figure 5: Find-RaTSS finds rationalizations in Diptheria and TB. Fig.(c)-(f) are our rationalizations (larger size by higher weight in r_j) in US map found by Find-RaTSS for the second and third cutpoint. Similar results for TB are also shown in Fig.(g)-(j).

Our rationalizations across c_1 gives actionable insights on the cities where Diptheria cases fluctuate the most after the campaign started. If we compare rationalization weights from Fig 5(c)-5(f) with Fig 5(g)-5(j) we observe, Find-RaTSS finds an interesting association between TB and Diptheria. r_j^u weights of the affected cities are positively correlated for both Diptheria and TB across the cutpoints. The same report (mentioned above) suggests this association is mainly due to presence of an iron-repressor gene. Clearly, by providing these insights, rationalizations can help DE understand the correlation between the diseases and design vaccination policies[28, 29].

4.3 Additional Case-study in general dataset (Wikipedia)

Here we provide a case-study showing that our algorithm can successfully identify culprits also in a general dataset even in a high dimensional time-series. To understand the importance of content and improve advertisement strategies, we consider the web traffic of 3000 Wikipedia articles from 2015 July-2017 October. The segmentation mark a different season of a year (around the end of 2015, middle of 2016, and beginning of 2017 in Fig. 6(a)). We find overall, the majority of articles/rationalizations (64%) by Find-RaTSS are eventful. Fig. 6(b) shows an example of the rationalizations by Find-RaTSS across the first cutpoint c_1. We observe Find-RaTSS considers some low-traffic articles, e.g., 'World's largest companies by sector' within top 10 along with the high traffic ones. The reason is, 'Forbes Global 2000' for worlds largest companies was published around June 2016 after c_1[6]. This is not easy to find such low traffic easily by visualizing the time-series or using any other baselines from Table. 3 (as buried with other high traffic articles).



(a) Time-series with segmentation

Rationalizations for c_1	Avg. traffic across c_1
Calculator	6819
Ahrar Ah-shams	4589
Christopher Paul Neil	3850
88th academy awards	1813
World largest companies by sector	223
Joker Comics	142

(b) Rationalization

Figure 6: Find-RaTSS finds the rationalization from Wikipedia during the period 2015-2017.

5 RELATED WORK

In this section, we explore our closest line of research in time-series mining. We also discuss in brief some remotely related work in interpretable AI models. To the best of our knowledge, no methods have been proposed so far for identifying actionable *TOIs* that can work for any black-box segmentation on multivariate time-series.

Urban Analytics: Several urban analytics applications in energy and public health domain have been stated for past years [31, 32]. Oak Ridge National Laboratory (ORNL) has developed a real-time situational awareness tool Eagle-I [8] to monitor and analyze the nation's energy infrastructure. The ORNL EARSS team [6] has an automated model to take wind speed and location estimates provided by hurricane experts a geospatial assessment on the impact to the electric grid in terms of projected duration of the outage. [22] shows how GPS mobility data can imply a spatial spread of influenza-like infectious diseases. Various works have been discussed recently on understanding and quantifying the impact of Covid-19 from different types of Non-pharmaceutical (NPI) interventions and exit strategies taken in Europe and China. [17, 25]. [33] developed an interactive visualization tool to observe mobility and sociability trends in different regions in the US due to Covid-19. [30] analyzed on time-series Tuberculosis (TB) data to characterize the demographic and temporal trends of the impact of the disease. [10] designed a change point detection process to detect the transition in multidimensional environmental crowdsensing data.

[5]https://timelines.issarice.com/wiki/Timeline_of_diphtheria

[6]https://www.forbes.com/sites/forbespr/2016/05/25/forbes-14th-annual-global-2000-the-worlds-biggest-public-companies-2/#21ca87643c44

Time-series mining: For explaining time-series classification models, [27] proposed how to discover the significant characteristic pattern of time-series using TF-IDF approach in vector space. Karlsson et al. developed an algorithm to find out the minimum number of tweaks in time-series data that can change a classifier decision like random shapelet forest classifier [14]. Jain et al. [13] proposed a problem to identify repeated sequences of pattern or motifs in time-series segmentation. There has been a couple of works on time-series segmentation algorithm to fulfill the objective in various domains [15]. Chen et al. designed an algorithm to automatically segment the sequences of any multivariate data without any prior knowledge of data distribution using information bottleneck principle [7]. Multivariate time-series segmentation algorithms, such as using distance-based measure [11] on domain- agnostic data, correlation network and variational EM [12] on automobile sensors, multilevel Hidden Markov Model (HMM) [21] on motion capture data, temporal mixture model [26] on railway data, and Kalman filters [19] on motion capture and chlorine data. [18] proposed a deep-learning approach for segmentation which automatically learns the features of the time-series using Autoencoder and detects a cutpoint when the difference of the features between two consecutive windows reach local maximum. Recently a novel spatio-temporal joint segmentation and explanation model has been proposed to identify failure phases in cyber-physical data [23]. However their explanation can only explain based on the internal state change of their segmentation model and their explanation model assumes each cutpoint in the segmentation as independent.
Interpretable AI models: Recently, various explainability models have been proposed to analyze influence of input features locally on model agnostic Machine Learning (ML) classifiers [16, 24]. [20] proposed to measure importance of instances considering the Shapley global effect of the model prediction. [9] first proposed AI rationalization to interpret autonomous agents on a game environment in terms of natural language entirely different from rationalization on time-series segmentation. On time-series there are several works on discovering characteristics of pattern [13, 27] and finding minimum number of tweaks to change a classifier decision [14].

6 CONCLUSION

In this paper, we introduce a novel problem Rationalizing time-series segmentation in terms of constituent time-series (RaTSS), to identify actionable time-series of interests for urban domain experts in a set of events found by time-series segmentation algorithms. We propose an algorithm Find-RaTSS to solve RaTSS in terms of a novel graph optimization problem using a segment graph data structure. Find-RaTSS successfully finds *TOIs* in several domains such as emergency management and the recent COVID-19. In addition we compare its performance with non-trivial baselines using synthetic and real-life general datasets with inferred ground-truth. As future work, we plan to explore other formulations for rationalizations like the average longest path instead of the longest path on the segment graph. We also intend to explore more complex *TOIs* (like scoring *groups* of time-series instead of individual ones as we do) which may be more suitable for some other applications.

7 ACKNOWLEDGEMENT

This paper is based on work partially supported by the NSF (Expeditions CCF-1918770, CAREER IIS-2028586, RAPID IIS-2027862), Medium (IIS-1955883, IIS-2106961), NRT DGE-1545362, and ORNL. We also thank all the reviewers, whose comments and suggestions helped to improve the manuscript.

REFERENCES

[1] [n. d.]. Web Traffic Time Series Forecasting. https://www.kaggle.com/c/web-traffic-time-series-forecasting/data.
[2] 2017. Georgia Power Working to Restore Service to 161,000 in DeKalb. https://www.dekalbcountyga.gov/news/georgia-power-working-restore-service-161000-dekalb.
[3] 2019. Flooding Hits Texas Towns Devastated by Harvey. https://www.nytimes.com/2019/09/19/us/houston-beaumont-flooding-imelda.html.
[4] 2021. Appendix. https://www.dropbox.com/s/jl11zsz4lvo2jzu/ratss_appendix_cikm2021.pdf?dl=0.
[5] 2021. Code repo. https://github.com/AdityaLab/RaTSS.
[6] Alan M Barker, Eva B Freer, Olufemi A Omitaomu, Steven J Fernandez, Supriya Chinthavali, and Jeffrey B Kodysh. 2013. Automating natural disaster impact analysis: An open resource to visually estimate a hurricane's impact on the electric grid. In *2013 Proceedings of IEEE Southeastcon*. IEEE, 1–3.
[7] Liangzhe Chen, Sorour E Amiri, and B Aditya Prakash. 2018. Automatic Segmentation of Data Sequences. AAAI.
[8] Eagle. 2012. Eagle-I. https://eagle-i.doe.gov/.
[9] Upol Ehsan, Brent Harrison, Larry Chan, and Mark O Riedl. 2018. Rationalization: A neural machine translation approach to generating natural language explanations. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. ACM, 81–87.
[10] Hafsa El Hafyani, Karine Zeitouni, Yehia Taher, and Mohammad Abboud. 2020. Leveraging Change Point Detection for Activity Transition Mining in the Context of Environmental Crowdsensing. *Proceedings of 9th International Workshop on Urban Computing ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020).
[11] Shaghayegh Gharghabi, Yifei Ding, Chin-Chia Michael Yeh, Kaveh Kamgar, Liudmila Ulanova, and Eamonn Keogh. 2017. Matrix Profile VIII: Domain Agnostic Online Semantic Segmentation at Superhuman Performance Levels. In *ICDM*. IEEE, 117–126.
[12] David Hallac, Sagar Vare, Stephen Boyd, and Jure Leskovec. 2017. Toeplitz inverse covariance-based clustering of multivariate time series data. In *KDD*. ACM, 215–223.
[13] Saachi Jain, David Hallac, Rok Sosic, and Jure Leskovec. 2018. CASC: Context-Aware Segmentation and Clustering for Motif Discovery in Noisy Time Series Data. *arXiv preprint arXiv:1809.01819* (2018).
[14] Isak Karlsson, Jonathan Rebane, Panagiotis Papapetrou, and Aristides Gionis. 2018. Explainable time series tweaking via irreversible and reversible temporal transformations. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 207–216.
[15] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. 2004. Segmenting time series: A survey and novel approach. In *Data mining in time series databases*. World Scientific, 1–21.
[16] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. *arXiv preprint arXiv:1703.04730* (2017).
[17] Shengjie Lai, Nick W Ruktanonchai, Liangcai Zhou, Olivia Prosper, Wei Luo, Jessica R Floyd, Amy Wesolowski, Mauricio Santillana, Chi Zhang, Xiangjun Du, et al. 2020. Effect of non-pharmaceutical interventions to contain COVID-19 in China. (2020).
[18] Wei-Han Lee, Jorge Ortiz, Bongjun Ko, and Ruby Lee. 2018. Time series segmentation through automatic feature learning. *arXiv preprint arXiv:1801.05394* (2018).
[19] Lei Li, James McCann, Nancy S Pollard, and Christos Faloutsos. 2009. Dynammo: Mining and summarization of coevolving sequences with missing values. In *KDD*. ACM, 507–516.
[20] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*. 4765–4774.
[21] Yasuko Matsubara, Yasushi Sakurai, and Christos Faloutsos. 2014. Autoplait: Automatic mining of co-evolving time sequences. In *SIGMOD*. ACM, 193–204.
[22] Robert Moss, Elham Naghizade, Martin Tomko, and Nicholas Geard. 2019. What can urban mobility data reveal about the spatial distribution of infection in a single city? *BMC public health* 19, 1 (2019), 1–16.
[23] Nikhil Muralidhar, Anika Tabassum, Liangzhe Chen, Supriya Chinthavali, Naren Ramakrishnan, and B Aditya Prakash. 2020. Cut-n-Reveal: Time Series Segmentations with Explanations. *ACM Transactions on Intelligent Systems and Technology*

(TIST) 11, 5 (2020), 1–26.

[24] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *KDD*. ACM, 1135–1144.

[25] Nick Warren Ruktanonchai, JR Floyd, Shengjie Lai, Corrine Warren Ruktanonchai, Adam Sadilek, Pedro Rente-Lourenco, Xue Ben, Alessandra Carioli, Joshua Gwinn, JE Steele, et al. 2020. Assessing the impact of coordinated COVID-19 exit strategies across Europe. *Science* (2020).

[26] Allou Samé and Gérard Govaert. 2012. Online Time Series Segmentation Using Temporal Mixture Models and Bayesian Model Selection. *ICMLA* 1 (2012), 602–605.

[27] Pavel Senin and Sergey Malinchik. 2013. Sax-vsm: Interpretable time series classification using sax and vector space model. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*. IEEE, 1175–1180.

[28] Swapna Thorve, Mandy L Wilson, Bryan L Lewis, Samarth Swarup, Anil Kumar S Vullikanti, and Madhav V Marathe. 2018. EpiViewer: an epidemiological application for exploring time series data. *BMC bioinformatics* 19, 1 (2018), 449.

[29] Srinivasan Venkatramanan, Jiangzhuo Chen, Sandeep Gupta, Bryan Lewis, Madhav Marathe, Henning Mortveit, and Anil Vullikanti. 2017. Spatio-temporal optimization of seasonal vaccination using a metapopulation model of influenza. In *2017 IEEE International Conference on Healthcare Informatics (ICHI)*. IEEE, 134–143.

[30] Win Wah, Sourav Das, Arul Earnest, Leo Kang Yang Lim, Cynthia Bin Eng Chee, Alex Richard Cook, Yee Tang Wang, Khin Mar Kyi Win, Marcus Eng Hock Ong, and Li Yang Hsu. 2014. Time series analysis of demographic and temporal trends of tuberculosis in Singapore. *BMC Public Health* 14, 1 (2014), 1121.

[31] Scott L Zeger, Rafael Irizarry, and Roger D Peng. 2006. On time series analysis of public health and biomedical data. *Annu. Rev. Public Health* 27 (2006), 57–79.

[32] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5, 3 (2014), 1–55.

[33] Fan Zuo, Jingxing Wang, Jingqin Gao, Kaan Ozbay, Xuegang Jeff Ban, Yubin Shen, Hong Yang, and Shri Iyer. 2020. An Interactive Data Visualization and Analytics Tool to Evaluate Mobility and Sociability Trends During COVID-19. *arXiv preprint arXiv:2006.14882* (2020).